2º curso / 2º cuatr.

Grado Ing. Inform.

Doble Grado Ing. Inform. y Mat.

Arquitectura de Computadores (AC)

Cuaderno de prácticas.
Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Alberto Jesús Durán López
Grupo de prácticas: 1
Fecha de entrega:
Fecha evaluación en clase:

1. Ejercicios basados en los ejemplos del seminario práctico

2. En el primer ejemplo de ejecución en atcgrid usando TORQUE se ejecuta el ejemplo `HelloOMP.c` usando la siguiente orden: `echo 'hello/HelloOMP' | qsub -q ac`. El resultado de la ejecución de este código en atcgrid se puede en el seminario. Conteste a las siguientes preguntas:

   a. ¿Para qué se usa en `qsub` la opción `-q`?

      RESPUESTA: Para indicar la cola a la que enviamos el resultado del proceso

   b. ¿Cómo sabe el usuario que ha terminado la ejecución en atcgrid?

      RESPUESTA: Cuando no hay procesos en la cola de trabajos ac (comprobar con la orden qstat)

   c. ¿Cómo puede saber el usuario si ha habido algún error en la ejecución?

      RESPUESTA: Se puede ver en el fichero de extensión ".e" que devueve los errores. Si su valor es 0 es que no se ha producido ninguno

   d. ¿Cómo ve el usuario el resultado de la ejecución?

      RESPUESTA: Se puede ver en el fichero de extensión ".o" (devuelve la ejecución)

   e. ¿Por qué en el resultado de la ejecución aparecen 24 saludos "¡¡¡Hello World!!!"?

      RESPUESTA: Porque hay 24 nodos asignados al trabajo y se ejecuta un 'Hello World' por cada nodo.

2. En el segundo ejemplo de ejecución en atcgrid usando TORQUE el script `script_helloomp.sh` usando la siguiente orden: `qsub script_helloomp.sh`. El script ejecuta varias veces el ejecutable del código `HelloOMP.c`. El resultado de la ejecución de este código en atcgrid se puede ver en el seminario. Conteste a las siguientes preguntas:

   a. ¿Por qué no acompaña a al orden `qsub` la opción `-q` en este caso?

      RESPUESTA: Porque ya está especificado en el script:

      #Se asigna al trabajo la cola ac

      #PBS -q ac

   b. ¿Cuántas veces ejecuta el `script` el ejecutable `HelloOMP` en atcgrid? ¿Por qué lo ejecuta ese número de veces?

      RESPUESTA:

   c. ¿Cuántos saludos "¡¡¡Hello World!!!" se imprimen en cada ejecución? (indique el número exacto) ¿Por qué se imprime ese número?

      RESPUESTA: 24 veces. Porque hay ese número de hebras

3. Realizar las siguientes modificaciones en el `script` "¡¡¡Hello World!!!":

- Eliminar la variable de entorno `$PBS_O_WORKDIR` en el punto en el que aparece.

- Añadir lo necesario para que, cuando se ejecute el script, se imprima la variable de entorno `$PBS_O_WORKDIR`.

Ejecutar el script con estas modificaciones. ¿Qué resultados de ejecución se obtienen en este caso? Incorporar en el cuaderno de trabajo volcados de pantalla que muestren estos resultados.

RESPUESTA: Primero modificamos el script, queda como resultado:

helloomp.o44911 | script_helloomp.sh

```
#!/bin/bash
#Se asigna al trabajo el nombre helloomp
#PBS -N helloomp
#Se asigna al trabajo la cola ac
#PBS -q ac
#Se imprime información del trabajo usando variables de entorno de PBS
echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
echo "Id. del trabajo: $PBS_JOBID"
echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
echo "Nodo que ejecuta qsub: $PBS_O_HOST"
echo "Cola: $PBS_QUEUE"
echo "Nodos asignados al trabajo:"
echo "Workdir: $PBS_O_WORKDIR"
cat $PBS_NODEFILE #Se fija a 12 el no de threads máximo (tantos como cores en un nodo)
export OMP_THREAD_LIMIT=12
echo "No de threads inicial: $OMP_THREAD_LIMIT"
#Se ejecuta HelloOMP, que está en el directorio en el que se ha ejecutado qsub
for ((P=OMP_THREAD_LIMIT;P>0;P=P/2))
do
export OMP_NUM_THREADS=$P
echo -e "\nPara $OMP_NUM_THREADS threads:"

done

```

Después seguimos los pasos siguientes:

En la terminal de ssh ejecutamos: qsub script_helloomp.sh

```
sftp> put scrip
script_helloomp.sh        script_helloomp.sh~
sftp> put script_helloomp.sh
Uploading script_helloomp.sh to /home/E1estudiante8/hello/script_helloomp.sh
script_helloomp.sh                            100%  823     0.8KB/s   00:00
sftp> get he
helloomp.e44892     helloomp.e44911     helloomp.o44892     helloomp.o44911

sftp> get helloomp.o44911
Fetching /home/E1estudiante8/hello/helloomp.o44911 to helloomp.o44911
/home/E1estudiante8/hello/helloomp.o44911     100%  538     0.5KB/s   00:00
sftp> get helloomp.e44911
Fetching /home/E1estudiante8/hello/helloomp.e44911 to helloomp.e44911
/home/E1estudiante8/hello/helloomp.e44911     100%  340     0.3KB/s   00:00
sftp>
```

Como resultado final obtenemos:

script_helloomp.sh | helloomp.o44911

```
Id. usuario del trabajo: E1estudiante8
Id. del trabajo: 44911.atcgrid
Nombre del trabajo especificado por usuario: helloomp
Nodo que ejecuta qsub: atcgrid
Cola: ac
Nodos asignados al trabajo:
Workdir: /home/E1estudiante8/hello
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
atcgrid1
No de threads inicial: 12

Para 12 threads:

Para 6 threads:

Para 3 threads:

Para 1 threads:

```

Resto de ejercicios

4. Incorporar en el fichero .zip que se entregará al profesor el fichero `/proc/cpuinfo` de alguno de los nodos de atcgrid (atcgrid1, atcgrid2, atcgrid3), y del PC del aula de prácticas o de su PC. Indique qué ha hecho para obtener el contenido de `/proc/cpuinfo` en atcgrid.

RESPUESTA: En la terminal de ssh ejecuto: echo 'cat /proc/cpuinfo' | qsub -q ac , donde envio el contenido del archivo /proc/cpuinfo a la cola ac.

Para obtenerlo en el ordenador local, en la terminal de sftp ejecuto el comando: get STDIN.e43318

Ese archivo se obtiene de la ejecución anterior

Teniendo en cuenta el contenido de cpuinfo conteste a las siguientes preguntas (justifique las respuestas):

a. ¿Cuántos cores físicos y cuántos cores lógicos tiene el PC del aula de prácticas o su PC?

RESPUESTA:

8 núcleos lógicos (hay 8 entradas en cpuinfo)

4 núcleos físicos

b. ¿Cuántos cores físicos y cuántos cores lógicos tiene un nodo de atcgrid?

RESPUESTA:

24 núcleos lógicos (las entradas que hay en cpuinfo)

12 núcleos físicos: 6 cpu cores * 2 procesadores

5. En el Listado 1 se puede ver un código fuente C que calcula la suma de dos vectores y en el Listado 2 una versión con C++:

```
v3 = v1 + v2;  v3(i) = v1(i) + v2(i),  i=0,…N-1
```

Los códigos utilizan directivas del compilador para fijar el tipo de variable de los vectores (v1, v2 y v3). En los comentarios que hay al principio de los códigos se indica cómo hay que compilarlos. Los vectores pueden ser:

- Variables locales: descomentando en el código `#define VECTOR_LOCAL` y comentando `#define VECTOR_GLOBAL` y `#define VECTOR_DYNAMIC`
- Variables globales: descomentando `#define VECTOR_GLOBAL` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_DYNAMIC`
- Variables dinámicas: descomentando `#define VECTOR_DYNAMIC` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_GLOBAL`. Si se usan los códigos tal y como están en Listado 1 y Listado 2, sin hacer ningún cambio, los vectores (v1, v2 y v3) serán variables dinámicas.

Por tanto, se debe definir sólo una de las siguientes constantes: `VECTOR_LOCAL`, `VECTOR_GLOBAL` o `VECTOR_DYNAMIC`.

a. En los dos códigos (Listado 1 y Listado 2) se utiliza la función `clock_gettime()` para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. En el código se imprime la variable `ncgt`, ¿qué contiene esta variable? ¿qué información devuelve exactamente la función `clock_gettime()`? ¿en qué estructura de datos devuelve `clock_gettime()` la información (indicar el tipo de estructura de datos y describir la estructura de datos)?

RESPUESTA: En el código, la función clock_gettime() está contenida en la biblioteca <time.h>. Esta calcula el tiempo que tarda en realizar la suma de vectores. Para ello declara una variable de tipo double donde guarda el resultado de la diferencia del valor cgt1 y cgt2.

Para saber más acerca de esta función, ejecutamos en la terminal: man clock_gettime

Aquí obtenemos que la structura que se usa es la siguiente:

```
The  res  and  tp  arguments  are  timespec structures, as specified in
<time.h>:

    struct timespec {
        time_t   tv_sec;        /* seconds */
        long     tv_nsec;       /* nanoseconds */
    };
```

b) Escribir en el cuaderno de prácticas las diferencias que hay entre el código fuente C y el código fuente C++ para la suma de vectores.

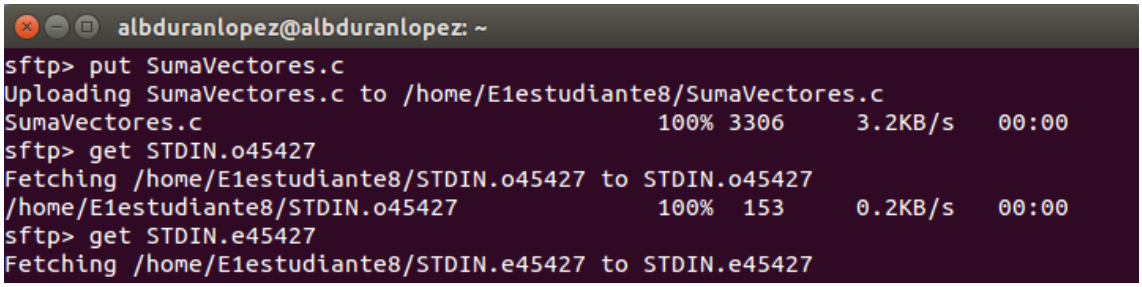
RESPUESTA:

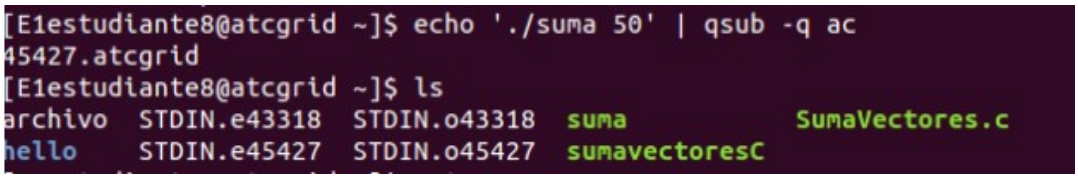
Descripción diferencia	En C	En C++
Distintas bibliotecas	stdlib.h,stdio.h,time.h	cstdlib, iosteam, time.h
Reserva de memoria	Se usa malloc	Se usa new double[N]
Salida	printf(“ mensaje “);	cout << “Mensaje”;
Liberar memoria reservada	free(v1)	Delete [] v1
Comprobación de si ha habido un error en la reserva de espacio para los vectores	Sí, línea 52	No

6. Generar el ejecutable del código fuente C del Listado 1 para vectores locales (para ello antes de compilar debe descomentar la definición de `VECTOR_LOCAL` y comentar las definiciones de `VECTOR_GLOBAL` y `VECTOR_DYNAMIC`). Ejecutar el código ejecutable resultante en atcgrid usando el la cola TORQUE. Incorporar volcados de pantalla que demuestren la ejecución correcta en atcgrid.

RESPUESTA:

Ejecutamos en la terminal de sftp el comando: put SumaVectores.c





Una vez ejecutado el código, obtenemos los resultados con la orden get de la primera captura.

El archivo STDIN.e45427 está vacio. Esto significa que no se ha producido ningún error.

El archivo STDIN.o45427 contiene lo siguiente:

Tiempo(seg.):0.000000563 / Tamaño Vectores:50 / V1[0]+V2[0]=V3[0] (5.000000+5.000000=10.000000) / / V1[49]+V2[49]=V3[49] (9.900000+0.100000=10.000000) /

Por lo que podemos afirmar que se ha ejecutado de manera correcta en atcgrid.

7. Ejecutar en atcgrid el código generado en el apartado anterior usando el `script` del Listado 3. Generar el ejecutable usando la opción de optimización –O2 tal y como se indica en el comentario que hay al principio del programa. Ejecutar el código también en su PC local para los mismos tamaños. ¿Se obtiene error para alguno de los tamaños? En caso afirmativo, ¿a qué se debe este error?

RESPUESTA: Ejecutamos los siguientes comandos desde la terminal de ssh y se obtiene error, violación de segmento por el tamaño de vectores tan altos que hay en el script:

N=65536;N<67108865;N=N*2

Como consecuencia se desborda la pila.

```
[E1estudiante8@atcgrid ~]$ gcc SumaVectores.c -O2 -o SumaVectoresC
[E1estudiante8@atcgrid ~]$ echo './SumaVectores.sh' | qsub -q ac
45453.atcgrid
[E1estudiante8@atcgrid ~]$ cat STDIN.e45453
./SumaVectores.sh: line 20: 18389 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 20: 18392 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 20: 18397 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 20: 18402 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 20: 18407 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 20: 18413 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 20: 18418 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
./SumaVectores.sh: line 20: 18423 Segmentation fault      (core dumped) $PBS_O_WORKDIR/SumaVectoresC $N
```

8. Generar los ejecutables del código fuente C para vectores globales y para dinámicos. Genere el ejecutable usando –O2. Ejecutar los dos códigos en atcgrid usando un `script` como el del Listado 3 (hay que poner en el `script` el nombre de los ficheros ejecutables generados en este ejercicio) para el mismo rango de tamaños utilizado en el ejercicio anterior. Ejecutar también los códigos en su PC local. ¿Se obtiene error usando vectores globales o dinámicos? ¿A qué cree que es debido?

RESPUESTA: En este caso no se obtiene error, no como en el ejercicio anterior ya que las variables no se almacenan en la pila y no se produce violación de segmento

9. Rellenar una tabla como la Tabla 1 para atcgrid y otra para el PC local con los tiempos de ejecución obtenidos en los ejercicios anteriores para el trozo de código que realiza la suma de vectores. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. Ayudándose de una hoja de cálculo represente en una misma gráfica los tiempos de ejecución obtenidos en atcgrid para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (eje x). Utilice escala logarítmica en el eje de ordenadas (eje y) en todas las gráficas. ¿Hay diferencias en los tiempos de ejecución con vectores locales, globales y dinámicos?

RESPUESTA:

Tabla 1 . Tiempos de ejecución de la suma de vectores para vectores locales, globales y dinámicos

Tiempos atcgrid

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.001155749	0.001744100	0.001473671
131072	1048576	0.002406484	0.003709740	0.002956282
262144	2097152	0.009751091	0.011794954	0.005821488
524288	4194304	Segmentation fault	0.015531513	0.041218363
1048576	8388608	...	0.031632379	0.023246563
2097152	16777216	...	0.130544048	0.045157744
4194304	33554432	...	0.166957545	0.092822544
8388608	67108864	...	0.351336533	0.183581949
16777216	134217728	...	0.500814248	0.398017117
33554432	268435456	...	1.041137960	1.029543313
67108864	536870912	...	1.041782464	1.500932755

Tiempos PC local

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	524288	0.000909834	0.000250060	0.001224419
131072	1048576	0.002310358	0.001574793	0.001268063
262144	2097152	0.003606552	0.002972109	0.004680237
524288	4194304	Segmentation fault	0.002530796	0.007535678
1048576	8388608	...	0.006006981	0.010281120
2097152	16777216	...	0.005895269	0.007941280
4194304	33554432	...	0.022204005	0.023133407
8388608	67108864	...	0.028526267	0.034378715
16777216	134217728	...	0.046070915	0.056080680
33554432	268435456	...	0.091792991	0.111979735
67108864	536870912	...	0.089767270	0.224582498

El tiempo para vectores locales es ligeramente más rápido que para el resto, sólo que a partir de 524288 nº de componentes se produce violación de segmento por lo que sólo es recomendable usarlos para un número de componentes no muy grande.
Comparando vectores globales y dinámicos, en general, los vectores globales son algo más rápidos.
Por último, vemos que los tiempos en nuestro PC local son bastante más pequeños que usando atcgrid.

10. Modificar el código fuente C para que el límite de los vectores cuando se declaran como variables globales sea igual al máximo número que se puede almacenar en la variable N (MAX=2^32-1). Generar el ejecutable usando variables globales. ¿Qué ocurre? ¿A qué es debido? Razone además por qué el máximo número que se puede almacenar en N es 2^{32}-1.

RESPUESTA:
No se puede compilar porque el archivo es muy grande. Se produce un error en la fase de enlazado, no en la de compilación. V1 sobrepasa el tamaño máximo y no queda espacio suficiente para V2 ni

V3


E1estudiante8@atcgrid:~
GNU nano 2.5.3 Fichero: SumaVectores.c Modificado
#include <time.h> // biblioteca donde se encuentra la función clock_gettime()
//#define PRINTF_ALL // comentar para quitar el printf ...
// que imprime todos los componentes
//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los$
//tres defines siguientes puede estar descomentado):
//#define VECTOR_LOCAL // descomentar para que los vectores sean variables ...
// locales (si se supera el tamaño de la pila se ...
// generará el error "Violación de Segmento")
#define VECTOR_GLOBAL// descomentar para que los vectores sean variables ...
// globales (su longitud no estará limitada por el ...
// tamaño de la pila del programa)
//#define VECTOR_DYNAMIC // descomentar para que los vectores sean variables ...
// dinámicas (memoria reutilizable durante la ejecución)
#ifdef VECTOR_GLOBAL
#define MAX 4294967295 //=2^32-1
double v1[MAX], v2[MAX], v3[MAX];
#endif
^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar txt^J Justificar^C Posición
^X Salir ^R Leer fich.^\ Reemplazar^U Pegar txt ^T Ortografía^_ Ir a línea


E1estudiante8@atcgrid:~
[E1estudiante8@atcgrid ~]$ gcc SumaVectores.c -o nuevo
/tmp/ccxMwA5b.o: En la función `main':
SumaVectores.c:(.text+0x105): reubicación truncada para ajustar: R_X86_64_32S co
ntra el símbolo `v2' definido en la sección COMMON en /tmp/ccxMwA5b.o
SumaVectores.c:(.text+0x14b): reubicación truncada para ajustar: R_X86_64_32S co
ntra el símbolo `v2' definido en la sección COMMON en /tmp/ccxMwA5b.o
SumaVectores.c:(.text+0x15d): reubicación truncada para ajustar: R_X86_64_32S co
ntra el símbolo `v3' definido en la sección COMMON en /tmp/ccxMwA5b.o
SumaVectores.c:(.text+0x1ce): reubicación truncada para ajustar: R_X86_64_32S co
ntra el símbolo `v3' definido en la sección COMMON en /tmp/ccxMwA5b.o
SumaVectores.c:(.text+0x1df): reubicación truncada para ajustar: R_X86_64_32S co
ntra el símbolo `v2' definido en la sección COMMON en /tmp/ccxMwA5b.o
SumaVectores.c:(.text+0x20a): reubicación truncada para ajustar: R_X86_64_PC32 c
ontra el símbolo `v3' definido en la sección COMMON en /tmp/ccxMwA5b.o
SumaVectores.c:(.text+0x212): reubicación truncada para ajustar: R_X86_64_PC32 c
ontra el símbolo `v2' definido en la sección COMMON en /tmp/ccxMwA5b.o
collect2: error: ld devolvió el estado de salida 1
[E1estudiante8@atcgrid ~]$